Cloud Computing for Comparative Genomics with Windows Azure Platform

Evolutionary Bioinformatics

TECHNICAL ADVANCE

Cloud Computing for Comparative Genomics with Windows Azure Platform

Insik Kim[1,2], Jae-Yoon Jung[1], Todd F. DeLuca[1], Tristan H. Nelson[3] and Dennis P. Wall[1,3]

[1]Center for Biomedical Informatics, Harvard Medical School, Boston, MA, USA. [2]School of Electrical and Computer Engineering, Ulsan National Institute of Technology, Ulsan, Korea. [3]Department of Pathology, Beth Israel Deaconess Medical Center, Boston, MA, USA. Corresponding author email: dpwall@hms.harvard.edu

Abstract: Cloud computing services have emerged as a cost-effective alternative for cluster systems as the number of genomes and required computation power to analyze them increased in recent years. Here we introduce the Microsoft Azure platform with detailed execution steps and a cost comparison with Amazon Web Services.

Keywords: cloud computing, Windows Azure, comparative genomics, Roundup, ortholog, orthologs, genomics, computational genomics

Evolutionary Bioinformatics 2012:8 527–534

doi: 10.4137/EBO.S9946

This article is available from http://www.la-press.com.

Introduction

As the number of sequenced genomes has been drastically increased over the last decade,[1] inadequate computational power has become a major bottleneck which hinders timely research in evolutionary bioinformatics. For example, currently there are more than 1800 sequenced species data available for comparative analysis,[2] and finding all common genes between any two different species that come from a single gene of the last common ancestor, referred to as orthologs,[3] will take more than 60 years of computation with a single-threaded, modern personal computer.[4] If a researcher aims to finish the ortholog computation in a week, he/she needs to obtain at least $60 \times 52 = 3120$ computing nodes for a whole week without interruptions. Although many research institutions now provide cluster or grid computing[5] services that enable researchers to execute multiple computing jobs in parallel, computing resources of this size may not be available because scalability of a system is limited by the total hardware capacity of the hosting institution, and many users have to share them in the same system.

In this context, cloud computing services have emerged as a cost-effective alternative for locally installed computing clusters. They provide computing resources and data storages that are virtually without limit, not interrupted by other users' applications or system maintenance, and charged by usage only. The remaining issues to migrate research applications currently running on a cluster based system to a cloud environment include which cloud service providers to choose and how difficult the learning curve for running a new cloud system would be. In order to provide a fair estimate on these issues, we have reviewed Amazon Web Services (AWS),[6–8] which is one of the main cloud platforms. Here we introduce another Windows-based cloud computing platform, Microsoft Azure, which is not yet as frequently used by the research community as AWS. We examine the details needed to run applications on the Azure platform with two case scenarios and compare its features with the elastic computing cloud offered by Amazon Web Services.

Methods and Technical Specifications

Windows® Azure™ platform

Azure platform virtualizes hardware resources and abstracts them into Virtual Machines (VMs). Each Azure application runs on one or more these VMs, just as if it is running on a dedicated computing machine. This abstraction layer above the physical hardware resources ensures scalability and portability, because any number of conceptually identical VMs can be readily added to or removed from an application. While Amazon Web Service provides "Infrastructure as a Service" (IaaS), meaning that users can make their own selection from the operating system (OS) level, Azure maintains Windows-based OS "Platform as a Service" (PaaS), so users do not need to manage or update the operating system by themselves.

Execution environment

An Azure application consists of one or more compute containers called "roles". A Web role, which deals with front end web communications, can be implemented with any of Internet Information Services (IIS) 7 compatible technologies, including ASP.NET, PHP, and Node.js. Secondly, a Worker role performs background tasks including data processing and supporting jobs for Web roles, and can host any type of application. Finally, a VM role can be used when OS-dependent platform configurations are necessary. It runs on a virtual image of a Windows Server 2008 R2OS, which can be configured locally then uploaded to the Azure platform. Each role may have multiple VMs or "instances" to actually perform jobs. For example, a typical Azure application may have a Web role with a single instance attached for hosting a web interface, a Worker role with several instances for data analysis, and communicating with the Web role through a message queue, and permanent storage to save analysis results.

Data storage

Each instance gets an allocation of volatile disk storage for temporary use. This storage is recycled during an instance reboot and does not persist after the instance is relocated for another job. For persistent data storage, the Azure platform currently provides four object abstractions:

1. BLOB stands for Binary Large OBject and is similar to files. Each storage account can have multiple containers, like directories in a Unix-based system, and each container has the actual BLOBs. Users can set a container as either public or private, and BLOBs in a public container can

be accessed by URL (e.g., http://<storagename>.blob.core.windows.net/<container>/<blob>). Accessing private container/BLOBs requires the storage account name and its access key.
2. Queues provide reliable storage and delivery of messages for an application to build loosely coupled and scalable workflow between Web roles and Worker roles.
3. Azure Table provides a table-structured storage mechanism based on the rows and columns format, and supports queries for managing the data. It is primarily aimed at scenarios where large volumes of data must be stored, while being easy to access and update.
4. SQL Azure Database is a scalable cloud database service built on SQL Server technologies, and supports the T-SQL based relational database model.

For enhanced security, data storage accounts and access keys are separately maintained from the main subscription accounts.

Networking services

A Content Delivery Network (CDN) allows users to cache publicly available static data for applications at strategic locations that are closer (in network delivery terms) to end users. The CDN uses a number of data centers around the world, which store the data in BLOB storage that has anonymous access.

Virtual Network Connect allows users to configure roles of an application running in Windows Azure and on computers on the local network so that they appear to be on the same network.

Requirements to Run an Azure Project

Subscription to the platform, account management, and project management can be done through the Azure project management portal (http://windows.azure.com). The subscription process requires a Windows Live ID, and the Silverlight™ plugin (http://www.silverlight.net/downloads) needs to be installed to display the portal site properly.

For rapid prototyping and development, Microsoft Visual Studio 2010 and Windows Azure SDK (Software Development Kit) for .NET (http://www.microsoft.com/web/gallery/install.aspx?appid=WindowsAzureToolsVS2010) are needed on a local machine. The Azure SDK includes local Compute and Storage Emulators where users can debug applications before deploying them to the cloud. Users can also perform some testing by running their applications on the Compute Emulator and by pointing the application at Azure storage in the test Azure environment.

Results

Case studies

Here we run our computationally intensive comparative genomics application, Roundup,[4] in two different case scenarios. First, we demonstrate how to run our application with a single Azure instance for prototyping and testing. Second, we show the scaled application with multiple instances with Azure HPC (High Performance Computing) Scheduler.

The Roundup application compares a pair of genomes from two different species based on the reciprocal smallest distance (RSD) algorithm. This algorithm consists of a series of computations, including NCBI BLAST, global sequence alignment (Kalign), maximum likelihood estimation of evolutionary distances (PAML), and a main Python script that controls the overall procedure. So in both scenarios, we put these programs in storage, install them, and run the script in each instance.

Running a single Azure instance for prototyping

In order to deploy an application into the Azure platform, users need to upload two files: the service configuration file (ServiceConfiguration.cscfg) that contains application configuration data, and the service package file that contains the rest of the application files, including service definition files, executables and scripts. Preparing, packaging, and uploading these files can be done using command line utilities like CSPack.exe, or using Visual Studio software. Here we use Visual Studio for rapid prototyping, as Visual Studio automatically generates default file templates when users create an Azure project and maintains the file and directory dependencies. A sample service configuration file used for this case is shown in Figure 1.

As the Roundup computation pipeline consists of multiple executables, we designed the Azure application to upload executables, and we incorporated the Python interpreter into Azure Storage, to write a startup script (startup.cmd) that downloads/installs programs,

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfiguration serviceName="Azure"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration"
osFamily="1" osVersion="*">
  <Role name="Worker">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
       value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

Figure 1. The service configuration file used in this case.
Note: Blue colored lines show a role property and its instance count.

and to execute the script when each computing instance starts to run. Invoking a script when an instance starts up can be done by specifying the script name on the service definition file (ServiceDefinition.csdef). A sample file used in this scenario is shown in Figure 2. Note that the script installs programs into the VM, so administrator privilege is needed to make it work properly. As specified in the file, the startup.cmd script should be placed on "Startup" folder under the "Worker" project.

Uploading an Azure application into the cloud can be done in two ways. First, Visual Studio enables users to publish their project after Azure account registration and storage setup. Second, the Azure project management portal provides an interface to define a new hosted service and to upload the corresponding configuration and package files from local machines. While it depends on the volume of a package and network bandwidth, uploading, deploying, and starting the hosted service application can take a significant amount of time. So debugging applications locally before deployment using the Compute and Storage Emulators is strongly recommended. After deployment, users can connect to any VM instance by Remote Desktop Service,[9] a feature provided by the Azure project management portal. In a remote desktop session, users can monitor performance of the connected instance by examining event logs and debug information generated from Azure diagnostics, a default imported module in each role.

Running a scaled application with Azure HPC scheduler

The second scenario is to run the Roundup application with multiple computing instances, using the Azure High Performance Computing (HPC) scheduler.[10] This plug-in supports parallel computing models including Message Passing Interface (MPI) and Service-Oriented Architecture (SOA), and provides built-in job scheduling and resource management features. For the purpose of our scenario, we only use the job management feature of the HPC scheduler. Users can submit any number of jobs to the scheduler, and the scheduler allocates and manages them in the compute nodes. The HPC scheduler, SDK, has a sample service template which consists of one head node, three compute nodes for Worker role, and one web frontend node for Web role. So a typical

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="Azure"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WorkerRole name="Worker" vmsize="Small">
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <Runtime executionContext="elevated" />
    <Startup>
      <Task commandLine="Startup\startup.cmd > c:\startup.log"
executionContext="elevated" taskType="simple">
      </Task>
    </Startup>
  </WorkerRole>
</ServiceDefinition>
```

Figure 2. The service definition file used in this case.
Note: Blue colored lines show a Worker role definition with elevated privileges, and specify startup.cmd as a startup task.

workflow with Azure HPC scheduler would be: (1) to deploy this service template first with or without modification; (2) to upload executables, input files, and scripts into Azure Storage; and (3) to submit a user job to the head node, which works as a HPC scheduler. Another difference from the previous scenario is that the HPC scheduler service keeps its own storage account and access keys for security purposes, so Storage information needs to be explicitly specified in all HPC commands or scripts. A modified version of the startup.cmd file for this case is shown in Figure 3.

The main Roundup execution script, run.cmd, downloads two genome input files from Storage, executes the RSD algorithm, and uploads the result file back to the output Storage container. This script is shown in Figure 4. Finally, Figure 5 shows an example of a job submission script, which can be run or input manually from the head node. The current job consists of one node preparation type task (i.e., run starup.cmd), Roundup computation tasks per each pair of genomes, and one node release type task that removes temporary data from each compute node. The "job submit" command finalizes the current job and submits it to run on the HPC cluster. More detailed information on the HPC job command can be found in the Azure command reference.[11]

Cost analysis

Azure platform charges pay-per-use, per-hour cost based on VM models as shown in Table 1.[12] Other charges include storage cost ($0.125 per GB stored/month, and $0.01/1000 storage transactions) and data transfer cost ($0.12/GB outbound transfer from Microsoft data centers in North America and Europe). Compared with Amazon EC2 pricing policy,[13] there is no significant cost difference between two providers. For example, large instance in the Amazon EC2 (2×2 GHz CPUs, 7.5 GB memory, and 850 storage[14]) matches best with large VM instance in the Azure platform. If we build a 100 node cluster using large instance from each provider, the Azure platform would cost $48 per hour, while AWS would charge $46 (or $32) per hour depending on Windows (or Unix/Linux) usage. In terms of storage and data transfer cost, the Azure platform and AWS charge the same amount up to 1 TB per month (storage) or up to 10 TB per month (data transfer).

Discussion

In this article, we examined whether the Windows Azure platform can readily support the computationally intensive task of orthology inference using the Roundup algorithm, in terms of computing capability and cost-effectiveness. In our test cases

```
cd /d %~dp0
set ACC=<STORAGE_ACCOUNT_NAME>              # Storage information can be obtained
set KEY=<STORAGE_ACCESS_KEY>                # After deployment of HPC scheduler.
set CNT=<CONTAINER_NAME>

start /w hpcpack download 7z.dll
/account:%ACC% /key:%KEY%                   # Download files from Storage
/container:%CNT%
start /w hpcpack download 7z.exe
/account:%ACC% /key:%KEY%
/container:%CNT%
start /w hpcpack download RSDpack.7z
/account:%ACC% /key:%KEY%
/container:%CNT%

start /w .\7z.exe x -y -oc:
c:\RSDpack.7z                               # Uncompress the main file

cmd /c "c:\RSDPack\install.cmd"             # Install programs
if not %errorlevel% equ 0 (echo "Install
Dependencies Fail at %0" && exit /b 12)     # Check if errors occurred during installation

start /w hpcpack download run.cmd
/account:%ACC% /key:%KEY%                   # Download the main execution script
/container:%CNT% /path:c:\RSDPack
exit /b 0
```

Figure 3. A modified version of startup.cmd script.
Note: It downloads Roundup pipeline components from Azure Storage using hpcpack command and installs them in each HPC compute node.

```
set ACC=<STORAGE_ACCOUNT_NAME>
set KEY=<STORAGE_ACCESS_KEY>
set CNT=<CONTAINER_NAME>
set OUT=<OUTPUT_CONTAINER_NAME>                                  # Set output container

set tmpDir=%CCP_WORKDIR%\task%CCP_JOBID%.%CCP_TASKID%            # Set tmp directory
mkdir %tmpDir% && cd %tmpDir%

start /w hpcpack download %1 /account:%ACC% /key:%KEY%           # %1 = 1st argument
/container:%CNT% /path:%tmpDir%
start /w hpcpack download %2 /account:%ACC% /key:%KEY%           # %2 = 2nd argument
/container:%CNT% /path:%tmpDir%

set OUT_FILE=%1_%2_0.8_1e-5.orthologs.txt                        # Set output file name

start /w python %CCP_WORKDIR%\RSD_win\build\scripts-             # Run RSD algorithm
2.7\rsd_search -q %1 -s %2 -o %OUT_FILE%

start /w hpcpack upload %OUT_FILE% /account:%ACC%                # Transfer output file
/key:%KEY% /container:%OUT%

cd .. && rmdir %taskDir% /s /q                                   # Clean up
```

Figure 4. The main Roundup execution script.
Note: This script takes two command—line arguments of genome file names, downloads them from Storage, and transfers the result file back to Storage after calculation is done.

using Roundup, the cross-platform migration from a Unix-based cluster system to a Windowsbased Azure platform did not raise significant issues, as most of the Azure- specific features, or APIs, can be accessed through Unix-style scripts. If users aim to transfer their Windows based applications such as those coded in ASP.NET into a cloud space, Azure would be one of the best candidates with the least modification of the current implementation. The main issue that we experienced while running the Roundup application in the cloud platform was that virtual machines are inherently stateless, meaning that users cannot get control over individual computing nodes, (for example turning on, suspending, or turning off

```
set ACC=<STORAGE_ACCOUNT_NAME>
set KEY=<STORAGE_ACCESS_KEY>
set CNT=<CONTAINER_NAME>

job new /jobname:RSD /projectname:Azure                          # Set a new job
set ID=2                                                         # JobId

job add %ID% /name:prep /type:NodePrep cmd /c
"hpcpack download startup.cmd /account:%ACC%                     # Run startup.cmd
/key:%KEY% /container:%CNT% /path:c: && call
c:\startup.cmd"

job add %ID% /workdir:c:\RSDpack run.cmd
Brucella_melitensis_biovar_Abortus.aa                            # Run main RSD script
Brucella_melitensis.aa
job add %ID% /workdir:c:\RSDpack run.cmd
Brucella_melitensis_biovar_Abortus.aa
Campylobacter_concisus_13826.aa
<…>

job add %ID% /name:release /type:NodeRelease cmd /c
"rmdir c:\RSDpack /s /q && if not %errorlevel% equ               # Release task
0 exit /b %errorlevel%

job submit /id:%ID%                                              # Submit job
```

Figure 5. Job submission script.
Notes: "job new" command returns a job id, which is 2 in this case. Adding tasks to the newly created job is done using this id.

Table 1. Azure platform pricing for virtual machine (VM) models.

Size	CPU cores	Memory	Volatile disk space	Bandwidth (Mbps)	Cost
Extra small	Shared × 1.0 GHz	0.75 GB	20 GB	5	$0.02
Small	1 × 1.6 GHz	1.75 GB	220 GB	100	$0.12
Medium	2 × 1.6 GHz	3.50 GB	490 GB	200	$0.24
Large	4 × 1.6 GHz	7.00 GB	1000 GB	400	$0.48
Extra large	8 × 1.6 GHz	14.00 GB	2040 GB	800	$0.96

Note: *For local storage resources in Web and Worker roles: 6,144 MB is reserved for system files.

individual nodes). Although the stateless property is essential for the scalability and portability of cloud services, many research-oriented tasks that require high throughput computing still consist of batch-based, independent processes that need to be completely redesigned for optimal cloud computing performance. A cost comparison with Amazon Web Services (AWS) showed that these two providers offer similar pay-per-use computing instance, storage, and data bandwidth models. However, AWS provides a more diverse set of instances from micro (for low throughput applications with periodic bursts) to graphics processing units (GPUs) based instances.

Going forward, we plan to explore methods to increase performance through a multi-core CPU architecture. In our previous study with Amazon Web Services, we showed that the overall cost for using a group of standard small instances is the same as using a group of High-CPU extra large instances to address a given problem.[6] While many other factors including I/O speed and memory size differences are involved in this result, we hypothesize that utilizing a multi-core architecture will increase the performance of High-CPU instances, which will therefore lower the overall cost.

Author Contributions

Conceived and designed the experiments: DPW Analyzed the data: ISK, JYJ. Wrote/edited the manuscript: DPW, ISK, JYJ, TFD, THN. Wrote the first draft of the manuscript: ISK. All authors contributed to, reviewed, and approved of the final manuscript.

Funding

This work was funded by grants from the National Science Foundation (0543480 and 0640809) awarded to DPW.

Competing Interests

Authors disclose no potential conflicts of interest.

Disclosures and Ethics

As a requirement of publication author(s) have provided to the publisher signed confirmation of com pliance with legal and ethical obligations including but not limited to the following: authorship and contributorship, conflicts of interest, privacy and confidentiality, and (where applicable) protection of human and animal research subjects. The authors have read and confirmed their agreement with the ICMJE authorship and conflict of interest criteria. The authors have also confirmed that this article is unique and not under consideration or published in any other publication, and that they have permission from rights holders to reproduce any copyrighted material. Any disclosures are made in this section. The external blind peer reviewers report no conflicts of interest.

References

1. Beiko RG. Telling the whole story in a 10,000-genome world. *Biol Direct*. 2011;6:34.
2. The UniProt Consortium. Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic Acids Res*. 2012;40(Database issue): D71–5.
3. Fitch WM. Distinguishing homologous from analogous proteins. *Syst Zool*. 1970;19(2):99–113.
4. DeLuca TF, Cui J, Jung JY, St. Gabriel KC, Wall DP. Roundup 2.0: enabling comparative genomics for over 1800 genomes. *Bioinformatics*. 2012;28(5): 715–6.
5. Bakery M, Buyyaz R. Cluster computing at a glance. In: Buyyaz R, ed. High Performance Cluster Computing: Architectures and System. Upper Saddle River, NJ: Prentice-Hall; 1999:3–47.
6. Kudtarkar P, DeLuca TF, Fusaro VA, Tonellato PJ, Wall DP. Cost-effective cloud computing: a case study using the comparative genomics tool, roundup. *Evol Bioinform Online*. 2010;6:197–203.
7. Fusaro VA, Patil P, Gafni E, Wall DP, Tonellato PJ. Biomedical cloud computing with Amazon Web Services. *PLoS Comput Biol*. 2011;7(8): e1002147.
8. Wall DP, Kudtarkar P, Fusaro VA, Pivovarov R, Patil P, Tonellato PJ. Cloud computing for comparative genomics. *BMC Bioinformatics*. 2010;11:259.

9. Using Remote Desktop with Windows Azure Roles [website], Washington: Microsoft Corporation; 2012. Available at: http://msdn.microsoft.com/en-us/library/windowsazure/gg443832.aspx. Accessed Jun 2012.
10. Windows Azure HPC Scheduler [website]. Washington: Microsoft Corporation; 2012. Available at: http://msdn.microsoft.com/en-us/library/hh560247(v=vs.85).aspx. Accessed Jun 2012.
11. Job subcommand reference [website]. Washington: Microsoft Corporation;, 2012. Available at: http://technet.microsoft.com/en-us/library/cc972848(v=ws.10).aspx. Accessed Jun 2012.
12. Microsoft Azure Pricing Overview [website]. Washington: Microsoft Corporation; 2012. Available at: http://www.windowsazure.com/en-us/pricing/details. Accessed Jun 2012.
13. Amazon EC2 pricing [website]. Seattle: Amazon.com, Inc. Available at: http://aws.amazon.com/ec2/pricing/. Accessed Jun 2012.
14. Amazon EC2 instance types [website]. Seattle: Amazon.com, Inc. Available at: http://aws.amazon.com/ec2/instance-types/. Accessed Jun 2012.